Tarea 4:
Árboles Cartesianos y Treaps

Nombre: Franco Zautzik

Email: francozautzik@gmail.com

Profesor/a: Patricio Poblete

Auxiliares: Gabriel Flores

Sven Reisenegger

Fecha de entrega: 16/07/2018

Introducción:

Los árboles cartesianos son una especie de árbol binario en el cual cada nodo almacena un par ordenado (x,y), x formando un arbol de busqueda binaria y y formando una árbol de prioridad (heap).

Para esta TDA se buscó programar la manera insertar en la ubicación correcta nuevos valores a este, imprimir en pantalla el árbol en postorden y finalmente entregar el costo promedio de una búsqueda exitosa en cada árbol. Además el programa debía sacar de un archivo de texto los valores (llave y prioridad) para los nodos y generar un árbol cartesiano a partir de ellos. Para formar el árbol simplemente se deben hacer inserciones como en un ABB común y luego hacer rotaciones simples hacia arriba hasta que el nodo a ingresar quedase bien ubicado.

En una segunda instancia se buscó hacer un experimento con treaps (árbol cartesiano donde se eligen al azar los valores de las prioridades de los nodos a la hora de insertarlos) en el que se buscaba conocer el promedio de los costos promedios de una búsqueda exitosa en un treap de n nodos.

Análisis del problema:

Teniendo en consideración el funcionamiento de los árboles cartesiano se buscó crear un programa implementado en java donde a partir de un archivo de texto, del cual sabemos la ubicación, se formará un árboles cartesiano con los datos que este contenía. La forma de los valores guardados en estos nodos sería de la forma (x,y) con x un entero que fuera la llave del nodo e y, un double, la prioridad de este.

El primer problema fue cómo insertar los nodos de manera tal que se cumpliera el orden de un árbol cartesiano. Los árboles cartesianos deben estar ordenados como ABB de acuerdo a la variable x y la prioridad y de un padre siempre debía ser mayor a la de sus hijos (en caso de tenerlos). Luego estaba el problema sencillo de cómo imprimir en postorden el árbol resultante y finalmente el cálculo del costo promedio de hacer una búsqueda exitosa en un árbol. Todo lo anterior se hizo bajo el supuesto de que no existieran valores repetidos de llaves y/o prioridades en los archivos de texto.

Posteriormente se estudió, utilizando la implementación de los problemas anteriores, el costo promedio de una búsqueda exitosa en un caso particular de árbol cartesiano, el treap. El treap es simplemente un árbol cartesiano donde se genera la prioridad de los nodos aleatoriamente a la hora de ingresar el nodo.

Solución del problema:

Para solucionar el primer problema de la inserción se creó el método insertar(x, y) que permite insertar los nodos de llave x y prioridad y respetando las reglas de ordenamiento en un árbol cartesiano. Para lograr esto primero se insertó los nodos siguiendo las reglas de un ABB tradicional donde se cumple que si la llave del nodo a insertar es mayor a la de la raíz, el nodo debía insertarse de manera recursiva en el hijo derecho y en el caso contrario en el hijo izquierdo. Luego en caso que fuera necesario se fueron haciendo rotaciones simples hacia arriba hasta conseguir que el nodo ocuparse el lugar correcto en el árbol de acuerdo a su prioridad. Para encontrar el lugar correcto se debe cumplir que su prioridad sea menor a la de su padre y mayor a la de sus hijos (en caso de tenerlos).

Luego está el problema de cómo imprimir el árbol resultante en postorden (con los nodos externos representados por "[]"). Para ello solo bastó notar que la impresión en esta notación está simplemente dada por la recursión del método imprimir:

```
//caso de un nodo exterior
if (AB == null){
          p = "[]" + p;
      }
//caso en el que se tiene un nodo interior
else{
          p = imprimir(AB.izq) + imprimir(AB.der) + "(" + AB.lla + "," + AB.prio + ")"
+ p;
      }
```

Posteriormente para calcular el costo promedio de una búsqueda exitosa el problema podía ser reducido a encontrar el costo total de todas las búsquedas exitosas en el árbol y la cantidad total de nodos en el árbol y luego simplemente hacer la división correspondiente al promedio.

En cuanto al experimento con los treaps, para llevarlo a cabo simplemente era necesario generar un loop de 1 hasta n en el cual se insertarán (con el método insertar(x, y)) las llaves (de 1 hasta n) con prioridades generadas aleatoriamente por la función Math.random(). A partir de los árboles generados anteriormente, y haciendo uso del método costoPromedio(), se calculó el costo promedio de hacer una búsqueda exitosa en el árbol resultante. Luego repitiendo 10 veces el mismo

experimento para ns fijos se calculó el promedio de los valores obtenidos para establecer el promedio de los costos promedios.

Modo de uso:

Es necesario par la utilización del programa que se establezca en el código (.java) la dirección del archivo de texto donde están guardados los nodos a ingresar en el árbol cartesiano utilizando la dirección del archivo en el computador siendo utilizado utilizando doble slash.

Ej: si el archivo solicitado está en la dirección “C:\Users\Bacchus\Desktop\Txt\ej1.txt” es necesario guardarlo de la forma “C:\\Users\\Bacchus\\Desktop\\Txt\\ej1.txt” en el String dir.

Además es importante notar que el experimento se lleva a cabo cuando de corre el programa independiente de el procedimiento de transformar los archivos de textos en árboles cartesianos.

Resultados:

Al utilizar el ejemplo dado en el enunciado en un archivo de texto se obtuvo el siguiente resultado para el arbol escrito en post orden:

[][](12,22.0)[](25,8.0)[][](42,17.0)[][][](78,15.0)(63,10.0)(56,6.0)[](90,5.0)(37,3.0)

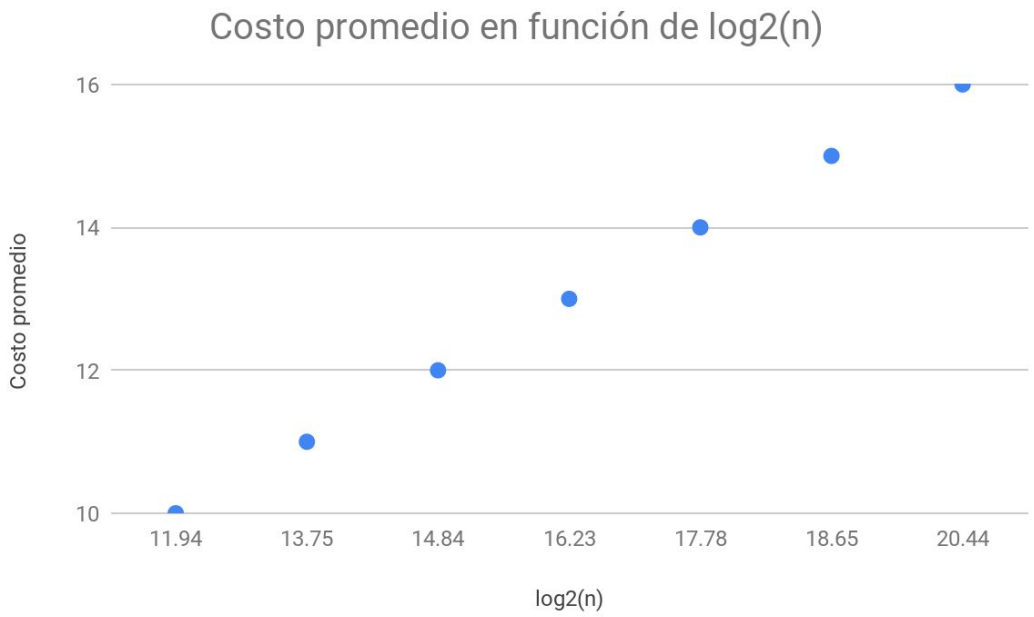
con el costo promedio de búsqueda exitosa:

3.0

lo cual corresponde al resultado esperado para la inserción de los valores, la impresión del árbol y el promedio de búsquedas en este árbol.

Para el experimento con los treaps se llegó a los siguientes resultados para distintos valores de n:

n	log2(n)	Promedio de los promedios del costo de una búsqueda exitosa
1024	10	11.94
2048	11	13.75
4096	12	14.84
8192	13	16.23
16384	14	17.78
32768	15	18.65
65536	16	20.44



Por lo que claramente el costo promedio tiene forma logarítmica en base 2. Lo anterior indica que para un caso aleatorio de búsqueda el promedio de los costos es menor a el número de nodos por lo que, como es el caso de una en un ABB, el costo promedio es menor que el orden n.